OULUN AMMATTIKORKEAKOULU

Eero Sova

OFFLINE-OMINAISUUDET OHJELMISTOKEHITYKSESSÄ

Offline toiminnallisuuksien kehitys Anicare-sovellukseen

OFFLINE-OMINAISUUDET OHJELMISTOKEHITYKSESSÄ

Offline toiminnallisuuksien kehitys Anicare-sovellukseen

Eero Sova
Opinnäytetyö
Kevät 2023
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen sv.

Tekijä(t): Eero Sova
Opinnäytetyön nimi: Offline ominaisuudet ohjelmistokehityksessä
Työn ohjaaja(t): Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Kevät 2023
Sivumäärä: 27

Opinnäytetyön aiheena oli offline-ominaisuuksien luominen Anicare-sovellukseen. Suoritin opinnäytetyön Anicare-yrityksessä. Anicare valmistaa Rudolf-laitteita, joita käytetään porojen sijainnin seuraamiseen. Porotalouden hoito tapahtuu pääasiassa Lapissa huonon verkon alueilla. Sovellukseen oli alun perin kehitetty vain online-tila, mutta siinä oli ongelmia huonon verkon alueella, esimerkiksi karttanäkymien lataus ja päivitys toimi hitaasti tai ei ollenkaan. Anicare-sovellus on kehitetty Ionic Angular -frameworkilla. Työn tavoitteena oli kehittää offline-toiminnallisuudet Anicare-sovellukseen.

Offline-toiminnallisuuksien luomiseen käytettiin QGIS-karttaohjelmaa, Ionic Angularia, OpenLayers-karttakirjastoa sekä SQLite-tietokantaa. Nämä ohjelmat, kirjastot sekä kehitysympäristöt olivat itselleni tuntemattomia, joten niiden toiminnan oppimiseen meni huomattava osa työajasta. Erityisesti sovelluksessa käytettävien offline-karttojen luominen QGIS-ohjelmassa oli vaikea toteuttaa, mutta lopputulos oli erittäin toimiva. Päädyin käyttämään rasteripohjaista karttaa vektoripohjaisen kartan sijasta, jota minua edeltänyt harjoittelija oli alkanut kehittämään. QGIS-ohjelmalla luodut kartat tuotiin sovellukseen käyttämällä OpenLayers-kirjastoa, jolla voi luoda karttanäkymiä Ionic-sovellukseen. SQLite-tietokantaa käytettiin käyttäjän sekä Rudolf-laitteiden tietojen tallentamiseen mobiililaitteessa ja näiden tietojen hakemiseen ohjelman käyttöliittymään. Kehityksessä käytettiin pääasiassa TypeScript-ohjelmointikieltä.

Kokonaisuutena opinnäytetyö oli onnistunut, tavoitteeksi asetetut ominaisuudet ohjelmassa tulivat valmiiksi. Näitä ominaisuuksia olivat sisäänkirjautuminen offline-tilassa, paliskuntakarttojen lataaminen sekä poistaminen mobiililaitteessa, paliskuntakarttojen näyttäminen Anicare-sovelluksen offline-näkymässä, offline- ja online-tilan välillä siirtyminen ohjelmassa ja paikallisen tietokannan luominen käyttämällä SQLite-tietokantaa. Nämä ominaisuudet toimivat pääasiassa hyvin, mutta jotkin ominaisuudet jäivät viimeistelemättä johtuen työajan loppumisesta

---

Asiasanat: Offline-ohjelmointi, Mobiili-ohjelmointi, Monialustakehitys, Ionic, Typescript, SQLite

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author(s): Eero Sova
Title of thesis: Offline features in software development
Supervisor(s): Jouni Juntunen
Term and year when the thesis was submitted: Spring 2023
Number of pages: 27

The topic of my thesis was creating Offline Features for the Anicare Application. I conducted the thesis work at Anicare, a company that manufactures Rudolf devices used for tracking reindeer locations. Reindeer husbandry primarily takes place in areas with poor network coverage. The original version of the application was designed for online use only, but it encountered issues in areas with weak network signals. For example, map views would load and update slowly or not at all. The Anicare application was developed using the Ionic Angular framework. The objective of the thesis was to develop offline functionality for the Anicare application.

To implement the offline features, I utilized the QGIS mapping software, Ionic Angular, the OpenLayers map library, and SQLite database. Since these programs, libraries, and development environments were unfamiliar to me, a significant portion of the work time was spent learning how they operate. Creating offline maps in QGIS software proved to be challenging but resulted in a highly functional outcome. I opted to use a raster-based map instead of the vector-based map that the previous intern had started developing. The maps created in QGIS were imported into the application using the OpenLayers library, which enables the creation of map views in Ionic applications. The SQLite database was used to store user and Rudolf device data on the mobile device and retrieve this information in the program's user interface. The development primarily involved TypeScript programming language.

Overall, the thesis work was successful, as the targeted features were implemented in the application. These features included offline login, downloading and deleting herding district maps on the mobile device, displaying herding district maps in the Anicare application's offline mode, transitioning between offline and online modes within the program, and creating a local database using SQLite. These features worked well for the most part, although some aspects remained unfinished due to the limited timeframe. Testing and bug fixing were incomplete, but the application functions well in offline mode.

Keywords: Offline programming, Mobile programming, Cross-platform development, Ionic, Typescript, SQLite

SISÄLLYS

1 JOHDANTO

Opinnäytetyö suoritetaan Anicare-yrityksessä. Anicare Oy tekee Rudolf-nimisiä poron seurantalaitteita sekä Anicare-sovellusta. Rudolf-laite asennetaan poron korvaan, ja laitteella on noin 10 vuoden elinikä. Rudolf-laite seuraa poron liikkumista ja lähettää käyttäjälle sijainnin. Viikoittain laitteella on myös mahdollista kerätä dataa poron käyttäytymisestä. Suurin osa poronhoitoalueista on ns. huonon verkon alueella, jossa matkapuhelinsovellusten internet-yhteydet ovat huonot tai niitä ei ole.

Anicare-sovelluksessa on kartta sekä laitteen jäljitysominaisuudet. Laitteen pääominaisuus on poron liikkeen seuraaminen sekä kuolinilmaisun lähettäminen, joka auttaa poron omistajaa löytämään kuolleen poron. Rudolf-laite lähettää käyttäjälle viikottain paikkasijainnin, joka näkyy Anicare-sovelluksessa. Käyttäjä voi myös pyytää poron sijaintitietoja tiheämmin halutessaan.

Aloitin Anicarella työskentelyn tehdessäni ammattiharjoittelua. Harjoitteluiden aikana aloitin Anicare-sovelluksen kehityksen ja tutustuin QGIS-ohjelmaan, jolla luodaan paliskuntakartat.

Harjoitteluiden sekä opinnäytetyön tavoite on luoda offline-tilassa toimivat kartat sekä siihen liittyvät ominaisuudet. Offline-tilassa tarvittavia ominaisuuksia ovat muun muassa sisäänkirjautuminen, paliskuntakarttojen luominen ladattavaan tiedostomuotoon, paliskuntakarttojen lataaminen, käyttäjän sijainnin näyttäminen, käyttäjän laitteiden eli porojen sijaintien näyttäminen sekä ladattujen paliskuntakarttojen näyttäminen offline-tilassa. Anicare-sovellus on suunniteltu pelkkä online-tila mielessä, joten offline-ominaisuuksien luominen pitää aloittaa aivan alusta ja muokata sovellusta niihin sopiviksi.

Offline-tilakehitykseen tarvitaan monenlaisia työkaluja. Karttojen luomiseen käytetään QGIS-ohjelmaa. QGIS on avoimen lähdekoodin karttasovellus. Sovellukselle luodaan tietokanta käyttämällä SQLite-tietokantaa. Anicare-sovellus on tehty Ionic Angular -frameworkilla, joten Ionic on iso osa projektia.

Opinnäytetyössä kehitetään vain offline-tilaan liittyviä ominaisuuksia, mutta Anicare-sovellusta kehitetään muiden toimesta samaan aikaan. Se voi luoda haasteita koodin yhteensopivuuden kanssa.

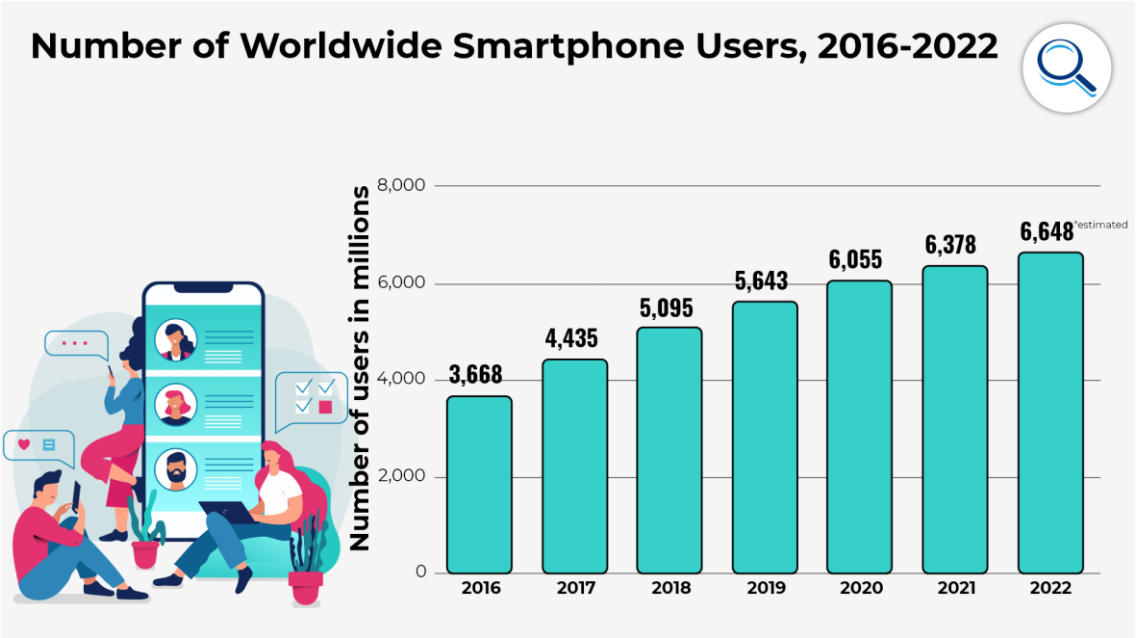
2 OFFLINE KEHITYS MOBIILILAITTEESSA

2.1 Mobiilikehitys

Mobiilikehitys on prosessi, jossa luodaan applikaatioita tai sovelluksia, jotka toimivat mobiililaitteissa kuten, älypuhelimissa, tableteissa sekä älylaitteissa. Sovelluksia kehitetään erilaisille käyttöjärjestelmille. Suosituimpia käyttöjärjestelmiä on iOS, Android ja Windows. Mobiilikehityksessä suunnitellaan ja kehitetään applikaatiota, suoritetaan testausta ja virheenjäljitystä ja lopuksi julkaistaan se esimerkiksi sovelluskauppaan. (Sam Solutions 2023.)

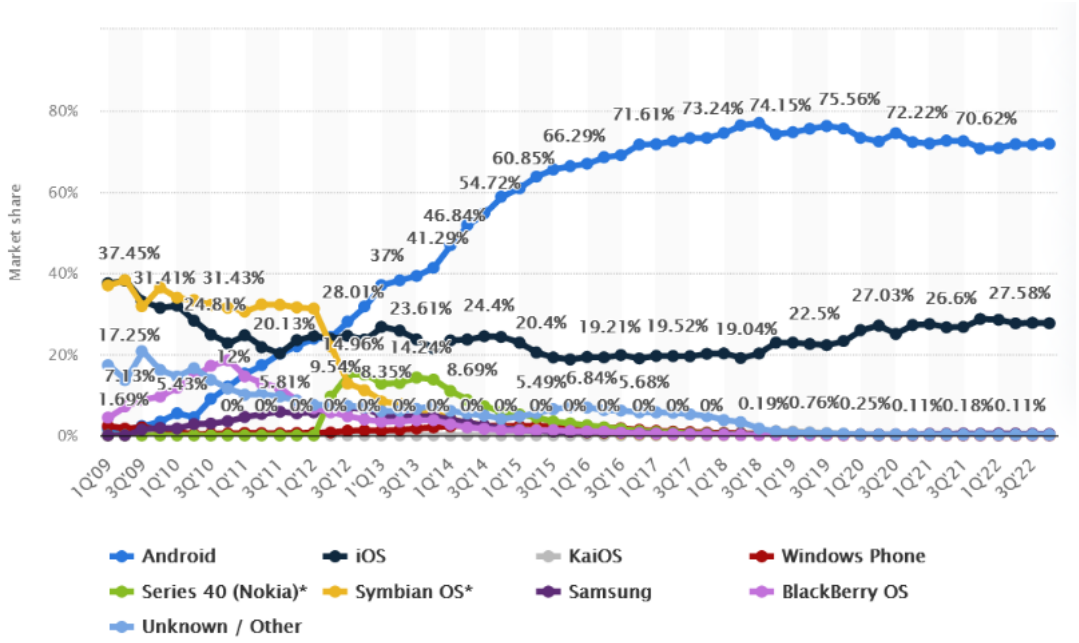
Mobiilikehitystä voidaan tehdä monilla erilaisilla ohjelmointikielillä ja kehitysalustoilla. Suosittuja ohjelmointi kieliä mobiilikehitykseen ovat Java, Kotlin, Swift, Objective-C ja C#. Kehitysalustat kuten React Native, Xamarin, Flutter ja Ionic tarjoavat monialustaisen lähestymistavan mobiilikehitykseen, joka mahdollistaa sovelluksien rakentamisen usealle käyttöjärjestelmälle käyttämällä yhtä koodikantaa. (Sam Solutions 2023.)

Mobiililaitteiden käyttäjien määrä on kasvanut jatkuvasti (kuva 1) ja nykyään ne ovat monille ihmisille päälaite, joten mobiilikehitys on tärkeä osa nykyajan sovelluskehitystä. Mobiilikehitykseen on luotu kehitysalustoja kuten Ionic, React Native, Flutter, Xamarin ja Swiftic. (InternetAdvisor 2023.)



KUVA 1 Älypuhelimien käyttäjämäärän kasvu vuosina 2016–2022 (InternetAdvisor 2023.)

Android ja iOS ovat käytetyimmät käyttöliittymät ja valtaosa kehityksestä on näiden käyttöliittymien parissa. Android on ylivoimaisesti suosituin käyttöliittymä, iOS on toiseksi suosituin ja yhdessä ne vastaavat 99,4 %;sta (kuva 2) mobiilimarkkinoiden kokonaisuudesta. (Monus 2023.)



KUVA 2. Älypuhelinkäyttöliittymien markkinaosuus vuosina 2009–2022. (Monus 2023.)

2.2 Natiivi kehitys

Mobiilikehityksessä natiivi (Native) tarkoittaa sovelluskehitystä, joka toteutetaan käyttäen mobiilialustan omaa ohjelmointikieltä ja -ympäristöä. Esimerkiksi iOS-alustan sovelluskehityksessä käytetään Swift- tai Objective-C-ohjelmointikieliä ja Xcode -kehitysympäristöä, kun taas Android-alustan kehityksessä käytetään Java- tai Kotlin-ohjelmointikieliä ja Android Studio -kehitysympäristöä.

Natiivin mobiilikehityksen hyötypuolia on parempi suorituskyky sekä käyttökokemus verrattuna hybridi-alustoihin, koska sovellukset on optimoitu kyseiselle alustalle. Se vaatii kehittäjältä osaamista alustan omassa ohjelmointikielessä sekä kehitysympäristössä, joka voi tehdä kehittämisestä haastavampaa. Jos sovelluksella on tarkoitus toimia usealla alustalla, natiivilla tehtäessä sovellus pitää rakentaa jokaiselle alustalle erikseen eli vaatii enemmän työtä kuin hybridillä kehittämine.

2.3 Ionic

Ionic on suosittu avoimen lähdekoodin kehitysalusta hybridisovelluksien kehitykseen, joka käyttää web-teknologioita, kuten HTML, CSS ja JavaScript. Ionicilla kehittäjä voi luoda mobiilisovelluksia, jotka toimivat iOS- ja Android-käyttöjärjestelmillä sekä verkkosivuna yhdellä koodikannalla. (Ionic 2023.)

Ionic tarjoaa laajan valikoiman ennakkoon rakennettuja UI-komponentteja ja teemoja, joilla kehittäjät voivat luoda moderneja, mukautuvia ja viehättäviä mobiili applikaatioita nopeasti. Kehitysalusta myös tarjoaa kehitystä helpottavan komentorivityökalun (CLI), joka yksinkertaistaa kehitysprosessia automatisoimalla yleisiä tehtäviä, kuten uusien projektien luomisen, sovelluksen rakentamisen ja eri alustoille julkaisemisen. (Ionic Framework 2023.)

Yksi Ionicin avain etu on mahdollisuus integroida suosittuja käyttöliittymän kehitysalustoja kuten Angular, React ja Vue.js. Tämän ansiosta kehittäjät pystyvät hyödyntämään näitä alustoja rakentaessaan laajennettavia sekä vakaita sovelluksia nopeasti ja helposti. Toinen Ionicin etu on sen laaja käyttäjäkunta, joka tarjoaa paljon resursseja ja tukea uusille sekä vanhoille kehittäjille. Ionic tarjoaa myös paljon liitännäisiä ja laajennuksia, joilla voi helposti integroida ominaisuuksia kuten kameran käyttöoikeuksia, ilmoituksia ja sisäisiä ostoksia. Kokonaisuutena Ionic on hyvä valinta kehittäjille, jotka haluavat rakentaa korkealaatuisia, monialustaisia mobiili sovelluksia nopeasti ja tehokkaasti käyttämällä tuttuja web-teknologioita. (Ionic 2023.)

2.3.1 Angular

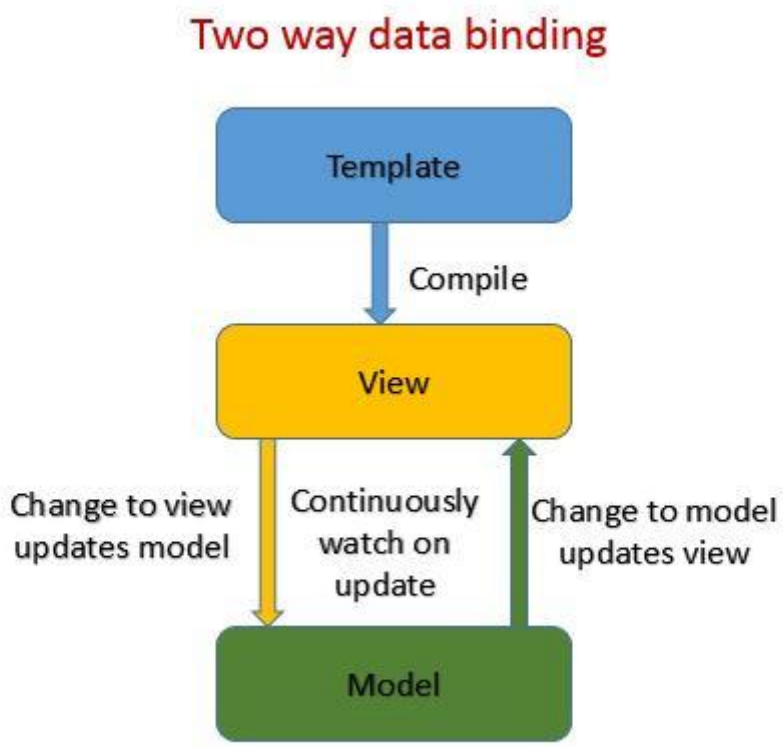
Angular on JavaScript sovellus kehitysalusta, jonka on luonut Google. Angular on komponenttipohjainen ohjelmistoalusta, joka käyttää TypeScript ohjelmointikieltä. Angularissa komponentit ovat sovelluksen rakennuspalikoita, jotka yhdistävät HTML-pohjaisen käyttöliittymän sekä TypeScript-pohjaisen logiikan. (Angular 2023.)

Angular sisältää myös ominaisuuksia kuten reitityksen, animaatiot, HTTP-palvelut ja testaus työkalut. Angular on avoimen lähdekoodin sovelluskehitysalusta, joten se on ilmainen käyttää ja sillä on paljon aktiivisia kehittäjiä, jotka tukevat Angularin kehitystä ja ylläpitoa. Angularista on tullut

suosittu sen tehokkuuden, laajennettavuuden, vakauden ja monipuolisuuden ansiosta. Se on yksi tärkeimmistä modernin sovelluskehityksen alustoista. (Angular 2023.)

Angular käyttää kaksisuuntaista tiedonsidontaa, joka mahdollistaa tietojen automaattisen päivittymisen sekä mallissa että näkymässä (kuva 3). Kaksisuuntaisella tiedonsidonnalla muutokset, jotka tehdään käyttöliittymässä, päivittyvät automaattisesti taustalla olevaan datamalliin ja päinvastoin. (Angular 2023.)

Kaksisuuntainen tiedonsidonta saavutetaan Angularissa yhdistämällä ominaisuussidonta ja tapahtumien sidonta. Ominaisuussidonta mahdollistaa datan sitomisen elementin ominaisuuteen, kun taas tapahtumasidonta mahdollistaa elementin tapahtumien käsittelemisen. (Angular 2023.)



KUVA 3. Kaksisuuntaisen tiedonsidonnan kuvaus. (Java Elegance 2023.)

Kaksisuuntainen tiedonsidonta voi helpottaa dynaamisten ja interaktiivisien verkkosovelluksien kehittämistä, sillä se poistaa tarpeen manuaaliselle synkronoinnille mallin ja näkymän välillä. Kuitenkin se voi johtaa suorituskykyongelmiin joissain tapauksissa, erityisesti suurissa sovelluksissa, joissa on monimutkaisia datamalleja. (Angular 2023.)

Angular käyttää Node Package Manageria (npm), joka on JavaScriptille tehty pakettienhallintaohjelma. Npm on maailman käytetyin pakettihallintaohjelma sekä ohjelmistorekisteri. Npm auttaa kehittäjiä asentamaan, päivittämään ja poistamaan paketteja helposti käyttämällä komentoriviä. (Npm 2023.)

2.3.2 Capacitor

Capacitor on avoimen lähdekoodin sovelluskehitysalusta, joka on suunniteltu monialustaisten sovellusten kehitykseen käyttämällä teknologioita kuten HTML, CSS ja JavaScript. Ionicin kehittäjät loivat Capacitorin vuonna 2018. Capacitor on nykyään korvannut Cordovan, joka oli ennen käytetyin kehitysalusta Ionicissa. Cordovan liitännäiset toimivat Capacitorissa. (Capacitor 2023.)
Yksi Capacitorin avain ominaisuuksista on sen kyky päästä käsiksi laitteen natiivi ominaisuuksiin yksinkertaisella ja yhtenäistetyllä rajapinnalla, joka helpottaa mobiilisovelluksien suunnitelua ja rakentamista web-teknologioilla. Näitä natiivi ominaisuuksia ovat esimerkiksi kamera, GPS, kiihtyvyysanturi ja kontaktit. Capacitor tukee monia alustoja yhdellä koodikannalla, näihin kuuluvat esimerkiksi iOS, Android ja web. Tämä helpottaa sovelluksen ylläpitoa ja päivittämistä sekä nopeuttaa kehitystä. (Ionic 2023.)

Capacitor myös tarjoaa lisäosa järjestelmän, jolla käyttäjä saa lisättyä alustaan lisäominaisuuksia tai kolmannen osapuolen kirjastoja. Tämä mahdollistaa ominaisuuksien kuten push-ilmoitukset, sovelluksen sisäiset ostokset ja ominaisuudet kuten analytiikan lisäyksen Capacitor sovelluksiin. (Capacitor 2023.)

Kaiken kaikkiaan Capacitor on joustava ja tehokas kehitysalusta monialustaisten sovelluksen kehittämiseen ja se tarjoaa monia etuja kehittäjille. Näihin etuihin kuuluu helppo pääsy laitteen natiivi ominaisuuksiin, monen alustan tuki ja lisäosa järjestelmä ominaisuuksien lisäämiselle. (Capacitor 2023.)

2.4 Offline-tallennus

Offline-tilassa toimiva sovellus tarvitsee paikallisesti tallennettuja tietoja, jotta sovellus toimii huonossa verkossa tai ilman verkkoyhteyttä. Yleisin tapa tallentaa tietoja paikallisesti on tietokanta.

Suosittuja tietokantoja sovelluskehityksessä ovat esimerkiksi SQLite, Realm ja CoreData (iOS). SQLite on kevyt ja nopea tietokanta, joka sopii erityisesti mobiilisovellusten kehitykseen. SQLite on maailman käytetyin tietokanta. (SQlite 2023.)

Sovellus voi myös online-tilassa ladata tiedostoja, joita se sitten käyttää offline-tilassa. Esimerkiksi sovellus lataa karttapaketin, jota se käyttää offline-tilassa kartta näkymän luomiseen. Tiedostojen lataaminen voi kestää pitkään, jos ne eivät ole pakattuna, joten yleensä tiedostot kannattaa ladata pakattuna ja sovelluksessa purkaa ne. (Microsoft 2023.)

Offline-sovelluksen tallennukseen on myös mahdollista käyttää välimuistia sekä paikallista tallennustilaa. Välimuisti nollaantuu kun sovelluksen käynnistää uudelleen eli se ei toimi yksin vaan tarvitsee esimerkiksi tietokannan, josta hakea tietoja. Paikalliseen tallennustilaan on mahdollista tallentaa yksinkertaisia tietoja. Androidilla on mahdollista käyttää SharedPreferences käyttöliittymää paikalliseen tallennukseen, Applen vastaava käyttöliittymä on UserDefaults.

Jotta sovellus toimii offline- ja online-tilassa samalla tavalla täytyy tietoja synkronoida näiden välillä. On monia tapoja tehdä synkronointi, esimerkiksi yksisuuntainen tai monisuuntainen. Yksisuuntaisessa tietoja lähetetään pelkästään laitteelta palvelimelle tai toisinpäin. Monisuuntaisessa palvelin sekä laite voivat lähettää ja vastaan ottaa tietoja.

Yleisiä tietojen synkronointi ratkaisuja ovat Firebase Realtime Database, Couchbase Mobile, PouchDB. Nämä tekevät offline-tallennuksesta sekä synkronoinnista suoraviivaista. Firebase Realtime Database sekä Couchbase Mobile ovat NoSQL tietokanta ratkaisuja. PouchDB käyttää Apache CouchDB:tä. (Firebase 2023; Couchbase 2023; PouchDB 2023.)

3 TOTEUTUS

3.1 Tausta

Opinnäytetyön tavoitteena oli tehdä Anicare-sovellukseen offline-ominaisuudet, joiden avulla sovellusta voitaisiin käyttää ilman internet-yhteyttä. Anicare-sovelluksen parissa työskentelyn aloitin yritysharjoittelussa.

Ensimmäisen harjoittelun tavoitteena oli luoda offline-tilassa käytettävät kartat, mikä osoittautui melko haastavaksi tehtäväksi. Edeltävä harjoittelija oli yrittänyt toteuttaa offline-karttaa vektorikartan avulla, mutta siinä oli ongelmia, kuten erilainen ulkoasu verrattuna online-tilan karttaan ja vaikeudet karttojen luomisessa. Päädyin itse käyttämään rasteripohjaista karttanäkymää tutkittuani erilaisia vaihtoehtoja.

Offline-kartat on luotu QGIS-ohjelmalla, joka on avoimen lähdekoodin karttasovellus. Ohjelmointitiimissä päätettiin QGIS-ohjelma, koska se oli maksuton vaihtoehto. Toisena vaihtoehtona olisi voinut olla ArcGIS-ohjelma, mutta se olisi ollut maksullinen, ja QGIS-ohjelmalle oli saatavilla enemmän ohjeita ja tukea, koska sillä on laaja kehittäjäyhteisö.

Aloin ensimmäisessä harjoittelussa tutustua QGIS:n toimintaan ja ominaisuuksiin. Aloin luoda offline-karttoja lataamalla Maanmittauslaitoksen karttoja tietokoneelle, mutta niiden käsittely osoittautui vaikeaksi. Lopulta päätin käyttää Maanmittauslaitoksen WMTS-palvelua (kuva 4), jolla sain haettua koko Suomen kartan QGIS-ohjelmaan.

KUVA 4. Maanmittauslaitoksen kartan ulkoasu. (Maanmittauslaitos 2023.)

Offline-karttojen luominen tapahtui käyttämällä Maanmittauslaitoksen WMTS (Web Map Tile Service) -palvelua. Yhdistämällä paliskuntien eli poronhoitoalueiden rajakoordinaatit ja WMTS-kartan

sain luotua kaikista Suomen paliskunnista karttanäkymät käyttämällä QGIS-ohjelmaa. Näitä tilelayereita pystyi sitten lisäämään Anicare-sovellukseen ja näyttämään offline-tilassa.

Seuraavaksi harjoittelun tavoitteena oli lisätä nämä tilelayerit Anicare-sovellukseen, mutta Anicare-sovellus oli suunniteltu toimimaan pelkästään online-tilassa. Tämän vuoksi lisätavoitteeksi tuli offline-ominaisuuksien kehittäminen Anicare-sovellukseen. Näitä ominaisuuksia olivat sisäänkirjautuminen offline-tilassa, karttojen lataaminen offline-tilassa, käyttäjän sijainnin sekä laitteiden sijainnin näyttäminen offline-tilassa, sovelluksen sisäinen navigointi eri näkymien välillä offline-tilassa sekä tietojen synkronointi offline- ja online-tilojen välillä. Ensimmäinen harjoittelu meni kokonaan QGIS:n opiskelussa sekä paliskuntakarttojen luomisessa.

Anicare-sovellus on luotu käyttäen Ionic Angularia, ja se ei ollut minulle entuudestaan tuttu. Seuraavien harjoitteluiden aikana opiskelin Ionic Angularin, Capacitorin sekä muiden kirjastojen toimintaa ja dokumentaatiota. Kun olin perehtynyt Ionicin ja Angularin toimintaan tarpeeksi, aloin kehittää offline-ominaisuuksia. Ensimmäiseksi kehitin offline-ominaisuudet kirjautumissivulle.

3.2 Implementointi

Opinnäytetyön suorittaminen alkoi tammikuussa Anicarella. Jatkoin siitä mihin harjoitteluissa olin jäänyt, ja työskentelin offline-ominaisuuksien parissa. Anicare-sovellusta kehitettiin samanaikaisesti muiden tekijöiden toimesta, joten koodiin tuli muutoksia, jotka piti ottaa huomioon offline-ominaisuuksia kehittäessä.

Kehitys alkoi offline-ominaisuuksien luomisesta sovellukseen. Jatkoin sisäänkirjautumissivun tekemistä. Tähän käytin Ionicin SQLite-kirjastoa. SQLite luo tietokannan laitteeseen, johon voi tallentaa ja hakea tietoja. Sisään kirjautuessa ohjelma tallentaa käyttäjän sisäänkirjautumistiedot tietokantaan, ja offline-tilassa se vertaa tietokannassa olevia tietoja annettuihin sisäänkirjautumistietoihin. Kun tiedot vastaavat tietokannassa olevia tietoja, sovellus siirtyy päänäkymään.

SQLite-komponentin nimi on Cordova SQLite Storage. Se on Ionicin Cordovalle luotu komponentti, jolla voi luoda tietokantoja ja lukea niitä. Komponentti toimii myös Capacitorilla, joka on nykyään käytännössä kokonaan korvannut Cordovan Ionic-kehityksessä. (Cordova SQLite 2023.)

Anicare-sovellus käyttää OpenLayers-kirjastoa karttojen luomiseen ja näyttämiseen. OpenLayers on suorituskykyinen ja ominaisuuksiltaan monipuolinen kirjasto interaktiivisten karttojen luomiseen verkossa ja mobiililaitteilla. Se pystyy näyttämään karttapaloja, vektoridataa ja merkintöjä, jotka on ladattu verkosta tai laitteen omista tiedostoista. (OpenLayers 2023.)

Myös offline-ominaisuuksien luomisessa Anicare-sovelluksessa päädyin käyttämään OpenLayers-kirjastoa, koska se oli jo valmiiksi käytössä sovelluksessa ja se tarjosi hyvin toimivia ratkaisuja tähän tarkoitukseen. OpenLayersin avulla pystyttiin luomaan karttanäkymiä, joihin voitiin lisätä vektoreita, rastereita ja merkintöjä. Yhdistämällä QGISillä luodut karttarasterit, paliskuntarajan vektorit ja laitteiden sijaintimerkinnät saatiin luotua offline-tilassa samannäköinen kartta kuin sovelluksen online-tilassa.

Seuraava tehtävä offline-ominaisuuksien kehittämisessä oli paliskuntakarttojen latausominaisuuden luominen. Jotta paliskunnat voitiin ladata sovellukseen myöhempää käyttöä varten, ne piti lähettää palvelimelle ja toteuttaa tiedostojen lataamisen toiminnallisuudet sovellukseen.

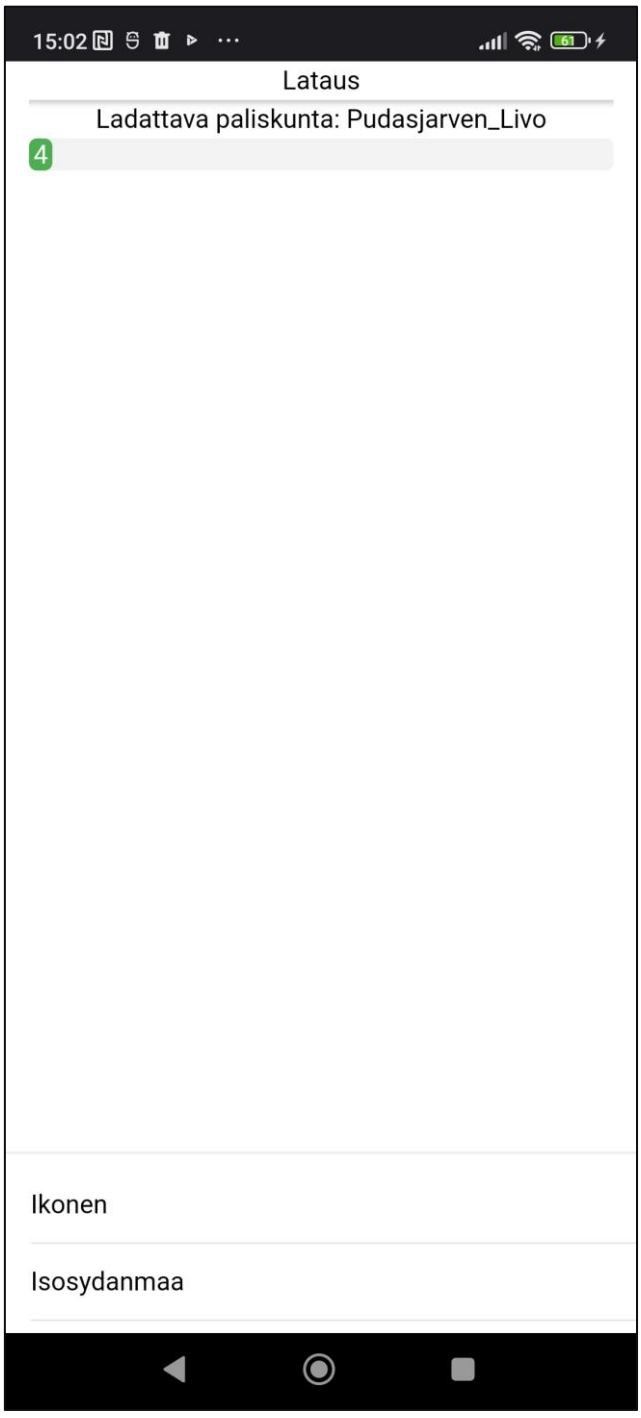
Tiedostojen latauksessa käytimme Cordova File Transfer -pluginia. Tämä plugin on Cordovalle kehitetty kirjasto, joka toimii myös Capacitorilla. File Transfer -pluginin avulla mobiililaitteelle voidaan ladata tiedostoja ja tallentaa ne haluttuun sijaintiin. Käyttämällä File Transfer -pluginia sovellus tallentaa paliskunnat mobiililaitteelle myöhempää käyttöä varten.

QGISin luomat kartat koostuivat useista pienistä kuvista, mikä hidasti niiden lataamista. Päätimme pakata tiedostot zip-tiedostoiksi. Jotta näitä pakattuja tiedostoja voidaan käyttää sovelluksessa, ne täytyy purkaa. Tiedostojen purkamiseen sovellus käyttää Cordovan Zip -pluginia (kuva 5), joka myös toimii Capacitorilla.

```
import { Zip } from '@ionic-native/zip/ngx';

constructor(private zip: Zip) { }

...

this.zip.unzip('path/to/source.zip', 'path/to/dest', (progress) => console.log('Unzipping, ' +
Math.round((progress.loaded / progress.total) * 100) + '%'))
 .then((result) => {
   if(result === 0) console.log('SUCCESS');
   if(result === -1) console.log('FAILED');
 });
```

KUVA 5. Esimerkki Cordova Zip- pluginin toiminnasta. (Ionic Docs 2023.)

Zip-pluginia käyttämällä on mahdollista seurata purkamisen edistymistä ja luoda palkki, joka näyttää käyttäjälle purkamisen etenemisen. Tämä palkki näkyy sovelluksen latausmodaalisivulla yhdessä latausliittymän kanssa. (Kuva 6.)

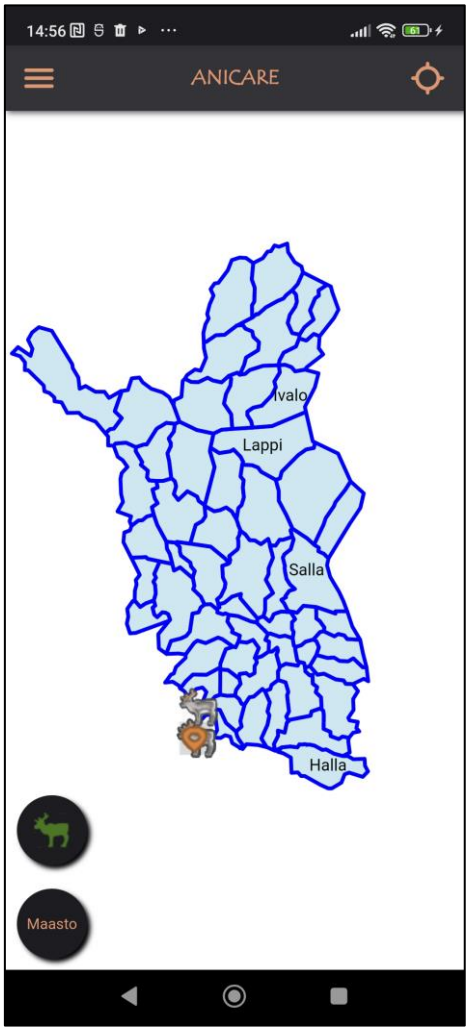


KUVA 6. Anicare sovelluksen kartan lataus UI.

Tämä kuva näyttää Anicare-sovellukseen luomani lataussivun. Käyttöliittymä on keskeneräinen eikä vielä täysin sovi sovelluksen yleiseen ulkoasuun, mutta pääpaino oli ominaisuuksien kehittämisessä eikä ulkoasun viimeistelyssä.

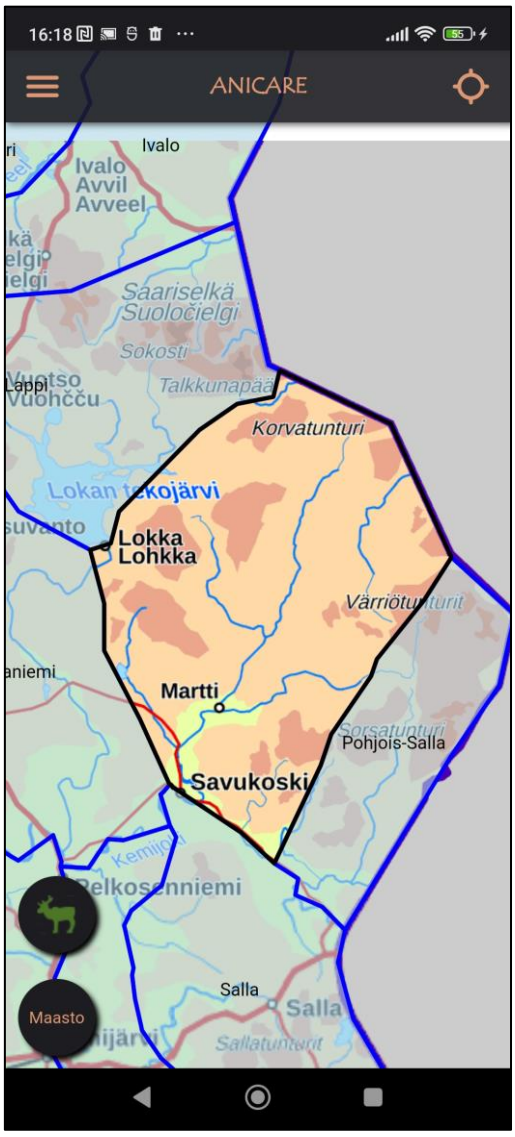
Yläosassa näkyy tällä hetkellä ladattavana oleva paliskunta, ja ruudun alareunassa näkyy jonossa olevat paliskunnat. Ohjelma lataa ja purkaa paliskunnat yksitellen. Kun paliskunta on ladattu ja purettu, siitä luodaan OpenLayersin avulla rasterikerros, joka näytetään offline-karttanäkymässä sovelluksen pääsivulla. Kun viimeinen paliskunta on ladattu, modaali sulkeutuu, ja ohjelma palaa takaisin paliskuntavalintanäkymään.

Sovelluksen pääsivulla joko näkyy offline- tai online-kartta, joka vaihtuu sovelluksen tilasta riippuen. Kuvassa näkyy sovelluksen ulkoasu (kuva 7), kun yhtäkään paliskuntaa ei ole vielä ladattu ja ollaan offline-näkymässä.



KUVA 7. Anicare-sovelluksen offline-karttanäkymä.

Kun paliskunta on ladattu laitteelle, sen alue muuttuu läpinäkyväksi, ja kun siirrytään paliskunnan rajojen sisäpuolelle karttanäkymässä, paliskunnan karttakerros tulee näkyviin. (kuva 8) Paliskuntien luomisen yhteydessä on asetettu maksimi tarkkuus, jotta niiden koko ei olisi liian suuri. Tiedostojen koko vaihtelee paliskunnan koon mukaan ja on noin 100–500 megatavua.



KUVA 8. Anicare sovelluksen offline-näkymä, näkyvissä Kemin-Sompio.

Offline-näkymän ja paliskuntakarttojen näyttämisen kehittäminen vei suuren osan opinnäytetyön työajasta. OpenLayers-kirjasto oli minulle uusi, ja sen oppimiseen kului aikaa. Lisäksi kohtasin

haasteita kehittäessäni logiikkaa useiden paliskuntakarttojen näyttämiseksi. Tavoitteena oli mahdollistaa useiden paliskuntien lataaminen laitteelle, mikä toi haasteita paliskuntien piilottamisen ja näyttämisen kanssa.

Päädyimme ratkaisuun, jossa vain yksi paliskuntakerros on näkyvissä käyttäjälle kerrallaan, ja muut ladatut paliskunnat piilotetaan. Tämä toteutettiin vertaamalla laitteen karttanäkymän keskipisteen Featurea ladattujen paliskuntien taulukkoon. (Kuva 9.)

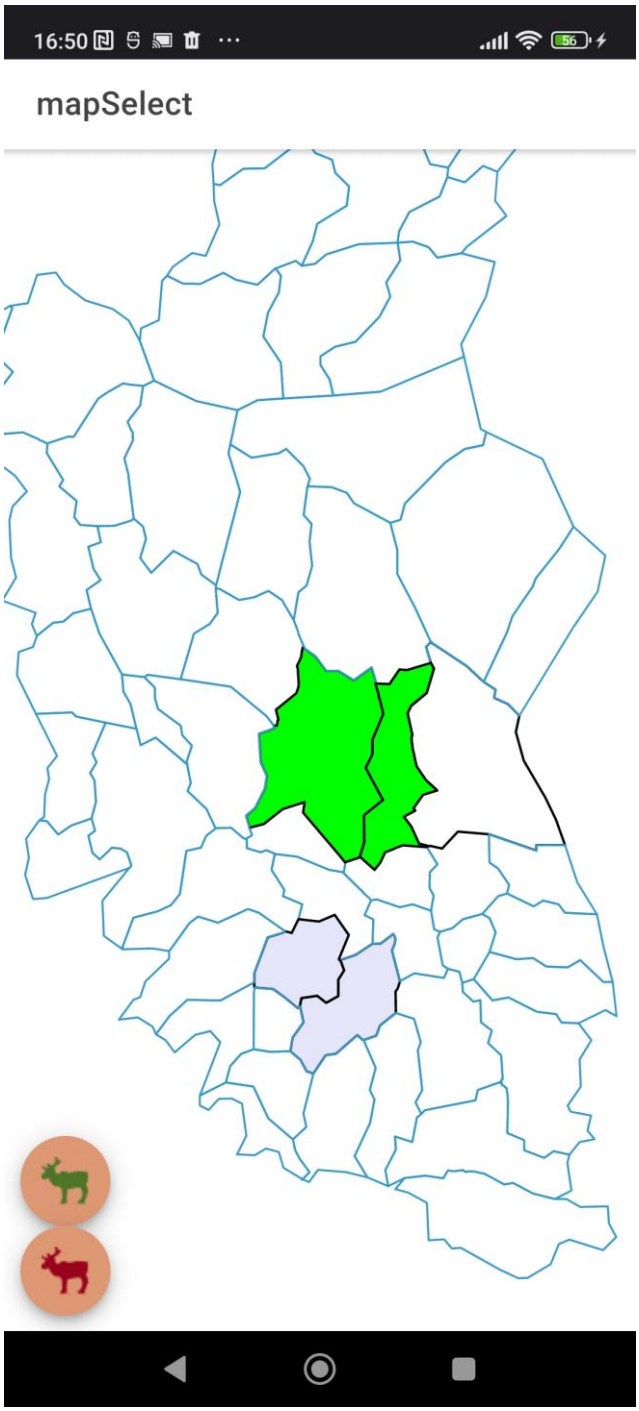
```
getCurrentLayer(location) {


   if (location.length !== 0) {
     if (location[0].get('Plk_nimi') !== this.oldLocation) {
       for (const value of this.arrayM.mapLayers) {
         if (location[0].get('Plk_nimi') === value.get('name')) {
           value.setVisible(true);
           this.activeMap = value.get('name');
         }

         else {
           value.setVisible(false);
         }
       }
     }
   }
 }
```

KUVA 9. Koodipätkä, joka vertaa kartta näkymän keskipistettä paliskunta taulukon kanssa.

Seuraavana tavoitteena oli luoda näkymä, jossa käyttäjä voi ladata haluamansa paliskunnat. Suunnittelun alkuvaiheessa oli useita ideoita, miten tämä voitaisiin toteuttaa. Ensimmäinen vaihtoehto oli luoda lista, jossa olisi kaikki Suomen paliskunnat, ja käyttäjä voisi valita haluamansa paliskunnat listalta. Toinen vaihtoehto oli luoda karttanäkymä, jossa olisi kaikki Suomen paliskunnat näkyvissä, ja käyttäjä voisi valita ja poistaa haluamiaan paliskuntia kartalta. Kolmas vaihtoehto oli yhdistää nämä ideat ja luoda kartta sekä lista.

Päädyimme käyttämään toista vaihtoehtoa, sillä se vaikutti parhaalta vaihtoehdolta. Kartalta on helpompi valita halutut paliskunnat kuin pitkältä listalta. Siksi loimme karttanäkymän, jossa näky-

villä on Suomen kartta ja paliskuntien rajaukset vektoreina. Käyttäjä voi valita haluamansa paliskunnat klikkaamalla niitä kartalla. Tässä näkymässä on myös mahdollisuus ladata ja poistaa paliskuntia. (Kuva 10.)



KUVA 10. Anicare sovelluksen paliskunta lataus näkymä.

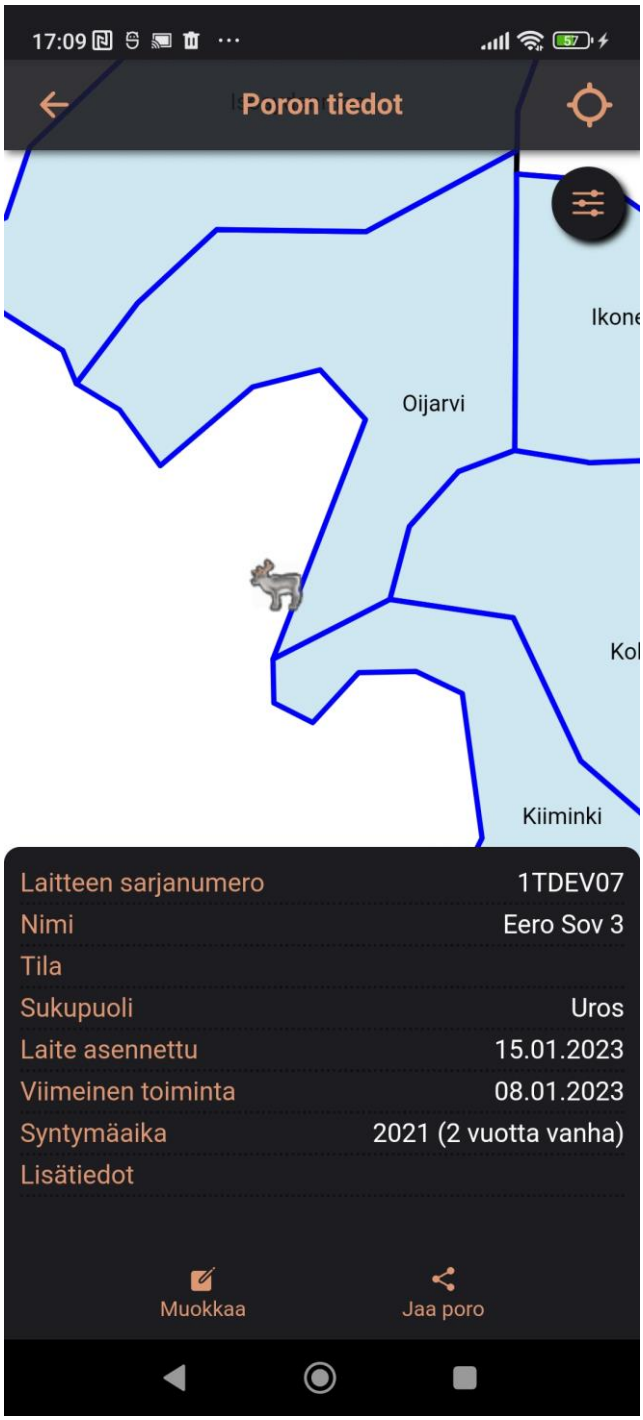
Tässä kuvakaappauksessa näkyy sovelluksen paliskuntalatauksen sivu. Käyttöliittymä on vielä keskeneräinen, mutta toiminnallisuudet ovat valmiita. Vaaleansinisellä merkityt paliskunnat on jo

ladattu sovellukseen, ja vihreällä merkityt paliskunnat ovat käyttäjän valitsemia. Vasemmassa alakulmassa näkyvät poro-ikonit ovat väliaikaisia painikkeita, joilla voidaan poistaa ja ladata paliskuntia. Punainen ikoni poistaa valitut paliskunnat, ja vihreä ikoni lataa valitut paliskunnat.

Kun käyttäjä painaa vihreää ikonia, sovellus siirtyy latausnäkymään (kuva 6) ja lataa paliskunnat. Kun paliskuntapaketit on ladattu, purettu ja paliskuntalayerit on luotu, sovellus palaa takaisin tähän kartan valintanäkymään. (Kuva 10.)

Viimeisenä haluttuna ominaisuutena oli näyttää käyttäjän ja laitteiden sijainti offline-näkymässä. Tämä osoittautui melko haastavaksi tehtäväksi, koska laitteelle piti tallentaa paljon poroihin liittyviä tietoja. Tähän tehtävään päätimme käyttää Cordova SQlite -pluginia. Tallennettavia tietoja olivat muun muassa poron ID, poron nimi, sarjanumero, laitteen akun kesto, viimeisen sijainnin leveys- ja pituusasteet, viimeisen saadun sijainnin aika sekä poron status (elävä/kuollut). Kaikki nämä tiedot haluttiin näyttää myös offline-näkymässä, joten ne piti tallentaa tietokantaan ja käyttää tietokannasta sovelluksessa.

SQLite oli minulle uusi, joten sen käyttämisen opetteluun kului jonkin verran aikaa. Jotta kaikki nämä tiedot saatiin helposti haettua laitteelle, piti luoda uusi API-kutsu, joka toi käytännössä kaikki laitteen tiedot sovellukseen. Näitä tietokannassa olevia tietoja hyödyntämällä saimme offline-tilassa näkymään samat tiedot kuin online-tilassa. (Kuva 11.)



KUVA 11. Offline-tilassa näkyvät laitteen tiedot.

Tämä ominaisuus oli yksi viimeisimmistä, jota kehitin opinnäytetyön aikana, ja se jäi hieman keskeneräiseksi. Sovelluksen ulkonäköä oli juuri muokattu muiden kehittäjien toimesta, joten Tila- ja Lisätiedot-kentät eivät ole vielä toiminnassa.

Tämän jälkeen kehitystyö siirtyi uusien ominaisuuksien kehittämisestä ohjelman testaukseen ja bugien korjaamiseen. Testatessa ilmeni, että sovelluksessa oli muistivuoto, mikä johtui käyttämästäni ohjelmointitavasta. Tämä onnistuttiin korjaamaan muuttamalla tapaa, jolla ohjelma välitti tietoja eri Ionic-palveluiden ja sivujen välillä. Suurempia ongelmia ei testauksessa ja vianetsinnässä löydetty.

Pääasiassa halutut offline-ominaisuudet saatiin valmiiksi, mutta muutaman sivun käyttöliittymä jäi keskeneräiseksi. Myös sovelluksen siirtyminen offline- ja online-tilojen välillä kehittämistyö jäi kesken. Suunnitelmissa oli käyttää Ionicin Network -pluginia havaitaksemme, milloin laitteella oli heikko verkkoyhteys, ja siirtyä tällöin offline-tilaan. Sovellusta kehittäessä käytimme on/off-toggle-nappia verkkotilojen välillä siirtymiseen, mutta se ei soveltunut julkaisuversioon.

4 POHDINTA

Opinnäytetyön tavoite oli luoda Anicare-sovellukseen offline-karttatoiminnallisuus. Tämä onnistui hyvin, vaikka aivan kaikki ominaisuudet eivät tulleet saavutetuiksi. Tämä johtui mielestäni tehtävän laajuudesta. Kaiken kaikkiaan toiminnallisuuksien luominen vaati monien kirjastojen sekä plug-inien käyttöä ja niiden toiminnan opettelua. Myös Ionic, Angular ja Capacitor olivat itselleni uusia teknologioita, joiden oppimiseen meni oma aikansa.

Työskentely opetti minulle paljon asioita ohjelmisto- ja mobiilikehityksestä. Jonkinlainen suunnittelu työn alussa olisi auttanut kehityksessä, mutta se olisi ollut vaikeaa tehdä tuntemattomalle alustalle. Opin paljon Ionicin, Angularin ja Capacitorin toiminnasta sekä siitä, miten ohjelmisto pitäisi kehittää näille alustoille. Myös suunnittelun tärkeys ohjelmistokehityksessä selveni minulle, ja jos useampi kehittäjä olisi työskennellyt offline-ominaisuuksien parissa, olisi suunnittelun ollut pakollista tapahtua kehityksen alussa.

Kirjastoja, joita käytin opinnäytetyön aikana, olivat erityisesti OpenLayers, SQLite, Zip, File ja File Transfer. Näiden käyttö opittiin hyvin, ja niistä on hyötyä tulevaisuudessa. Opin myös API-kutsujen luomisesta palvelinpuolella sekä ohjelmassa kutsun luomisesta ja sen tietojen käyttämisestä.

Kehityksessä olisi ehkä ollut mahdollista käyttää Firebasen Firestore-tyyppisiä ratkaisuja offline-tietokannan luomiseen. Tämä olisi mahdollisesti helpottanut kehitystä sekä offline- ja online-tietojen synkronointia, mikä oli haastavaa luoda SQLiteä käyttämällä. Tämä olisi kuitenkin vaatinut ohjelmaan suuria muutoksia verrattuna SQLite-ratkaisuun, jolla offline-tietokannan luominen onnistui ilman suuria muutoksia ohjelman kokonaistoimintaan tai palvelinpuolelle.

Kaiken kaikkiaan opinnäytetyö oli onnistunut, vaikka kaikki halutut ominaisuudet eivät olleet tulleet aivan täysin valmiiksi. Keskeneräiseksi jäi pääasiassa käyttöliittymä muutamilla sivuilla, mutta itse ominaisuudet ovat valmiita. Erityisesti karttojen luominen QGIS-ohjelmalla oli vaativa osuus, mutta rasterikarttojen luominen oli erityisen toimiva ratkaisu. Offline-karttaominaisuuden vieminen julkaisuvalmiiksi ei ole kovin suuri työ. Se vaatii käyttöliittymän viimeistelyn lataussivulla sekä laite sivulla, yleistä testausta ja muutamien virheiden korjaamista.

LÄHTEET

Angular 2023. What is Angular?. Hakupäivä 22.05.2023. https://angular.io/guide/what-is-angular

Capacitor 2023. Capacitor Plugins. Hakupäivä 09.05.2023 https://capacitorjs.com/docs/plugins

Cordova SQlite 2023. SQlite storage plugin for Cordova. Hakupäivä 19.05.2023. https://github.com/storesafe/cordova-sqlite-storage

Couchbase, 2023. Couchbase Mobile. Hakupäivä 02.05.2023. https://www.couchbase.com/products/mobile

Firebase, 2023. Firebase Realtime Database. Hakupäivä 02.05.2023. https://firebase.google.com/docs/database

Internet Advisor 2023.. Smartphone Statistics in 2023: Everything in Your Pocket. Hakupäivä 16.05.2023 https://www.internetadvisor.com/smartphone-statistics

Ionic 2023. Capacitor vs Cordova. Hakupäivä 09.05.2023 https://ionic.io/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development

Ionic docs 2023. Zip Angular. Hakupäivä 20.05.2023. https://ionic-docs.herokuapp.com/docs/native/zip

Ionic Framework 2023. Introduction to Ionic. Hakupäivä 09.05.2023 https://ionicframework.com/docs

Java Elegance 2023. Kuva. Two way data binding vs Traditional Approach in AngularJs with example. Hakupäivä 15.05.2023 http://javaelegance.blogspot.com/2015/11/two-way-data-binding-angularjs-example.html

Maanmittauslaitos 2023. Karttapaikka. Hakupäivä 20.05.2023. https://asiointi.maanmittauslaitos.fi/karttapaikka/

Microsoft 2023. Zip and unzip files. Hakupäivä 16.05.2023. https://support.microsoft.com/en-us/windows/zip-and-unzip-files-8d28fa72-f2f9-712f-67df-f80cf89fd4e5

Npm 2023. About npm. Hakupäivä 22.05.2023. https://docs.npmjs.com/about-npm

OpenLayers 2023. Overview. Hakupäivä 19.05.2023. https://openlayers.org/

PouchDB, 2023. PouchDB FAQ. Hakupäivä 02.05.2023. https://pouchdb.com/faq.html

Monus, Anna 2023. The 2023 guide to native app development. Hakupäivä 16.05.2023. https://ray-gun.com/blog/native-app-development/

Sam Solutions. 2023. Cross-Platform Mobile Development: Five Best Frameworks. Hakupäivä 09.05.2023 https://www.sam-solutions.com/blog/cross-platform-mobile-development/

SQLite. 2023. About SQLite. Hakupäivä 28.03.2023. https://sqlite.org/about.html.